논문 2008-45SD-1-1

비교기를 사용하지 않는 부호화-절대값 가/감산기 설계

(A Design of Comparatorless Signed-Magnitude Adder/Subtracter)

정 태 상*, 권 금 철**

(Tae-Sang Chung and Keum-Cheol Kwon)

요 약

이진수 시스템에서는 하드웨어 구현, 연산속도 등에 따라 음수와 양수를 나타내는 여러 가지 수 표현법이 있다. 그 중에서 한 비트로 부호를 정하고 나머지 비트들로 절대값을 표현하는 부호화-절대값 표현법은 간단하고 부호비트를 변환 시키는 것만으로 음수를 구할 수 있다. 그러나 부호화-절대값 표현법에서 실제 계산은 연산과 연산자들의 부호에 따른 절대값 비교를 필요로 한다. 간단한 구조에서 두 부호화-절대값 수의 덧셈, 뺄셈 연산기는 비교기와 선택적인 보수기, 덧셈기로 구성된다. 본 논문에서는 명시적인 비교기 사용 없이 두 수의 차이를 구할 수 있는 회로를 설계하고 이 회로를 이용하여 두 부호화-절대값으로 표현되는 수의 덧셈/뺄셈을 수행하는 가/감산기 설계하였다

Abstract

There are many possible representations in denoting both positive and negative numbers in the binary number system to be applicable to the complexity of the hardware implementation, arithmetic speed, appropriate application, etc. Among many possibilities, the signed-magnitude representation, which keeps one sign bit and magnitude bits separately, is intuitively appealing for humans, conceptually simple, and easy to negate by flipping the sign bit. However, in the signed-magnitude representation, the actual arithmetic operation to be performed may require magnitude comparison and depend on not only the operation but also the signs of the operands, which is a major disadvantage. In a simple conceptual approach, addition/subtraction of two signed-magnitude numbers, , requires comparator circuits, selective pre-complement circuits, and the adder circuits. In this paper circuits to obtain the difference of two numbers are designed without adopting explicit comparator circuits. Then by using the difference circuits, a universal signed-magnitude adder/subtracter is designed for the most general operation on two signed numbers.

Keywords : Signed-Magnitude Number Representation, Difference Circuits, Adder/Subtracter, Complement

I. 서 론

양수와 음수를 함께 표현하는 방법은 부호비트와 크기를 따로 가지는 부호화-절대값 표현법, 정해진 bias를 모든 수에 더해주는 bias 표현법, 음수를 보수로 표현하는 보수표현법, 그리고 수의 각 자리에 부호를 따로 붙이는 부호화 표현법 등이 있다. 모든 양수/음수 표현법은 하드웨어 구현의 복잡성, 연산 알고리즘, 연산속도 등에 있어서 각각의 장점과 단점을 함께 가지며, 각각의 적용 분야도 이 장단점에 따라 다르게 된다. 이 중에서 부호화-절대값 표현법은 인간의 관점에서 직관적으로 쉽게 받아들일 수 있고, 개념적으로 간단하며, 양수와 음수의 범위가 같으며, 그리고 사인 비트만 뒤집으면 되는 부호 변경이 용이한 장점들을 가진다. 그렇지만, 부호화-절대값 표현법의 주 단점은 가감산에 있어서 실제로 행하는 연산은 두 수의 연산과 부호에 따라 달라질 수 있는 점이다.

일반적인 음수, 양수의 감산의 경우 A-B는 B의 2의 보수를 취하여 더하는 방식 즉 $A+\overline{B}+1$이 효율적이다. 그러나 2의 보수로 표현된 수 A와 B(A와 B는 자체적으로 부호를 가지고 있음)의 통합 가감산, 즉

* 정회원, ** 학생회원 중앙대학교 전자전기공학부
(Dept of Electrical and Electronics Engineering, University of Chung-Ang)

※ 이 논문은 2004년도 중앙대학교 학술연구비 지원에 의한 것임
접수일자: 2007년7월5일, 수정완료일: 2008년1월7일

$\pm(A)\pm(B)$ 중의 어떤 경우에는 A와 B 두 수 모두 2의 보수를 취해야 할 필요가 있다. 이 경우에는 추가적인 보수(complememnt)회로나 1증가 회로의 지연이 따르게 된다. 부호화-절대값으로 표현되는 수의 가/감산의 경우, 두 수의 부호와 연산에 따라 실제 행하는 연산이 달라지게 되고 이때 두 절대치의 대소비교가 선행되어야 한다.

부호화-절대값 수 표현법은 총 자리수를 n+1이라고 할 경우 최상위의 1 bit은 0과 1로써 각각 양수와 음수를 표시하고 나머지 n 자리의 수로서 크기를 표시하는 방법이다. 따라서 양수와 음수의 범위는 각각 0부터 2^n-1까지 2^n개 인데, 이중 0에 대하여는 ±0이 존재하게 된다. 이런 수 표현법은 개념적으로 간단하여 제어공학 등에서 채집되는 원시 데이터의 표현으로 적합하고, 부동소수점 표현법에서 지수부분을 표현하는 방법으로 많이 사용된다.[1]

부호화-절대값 표현법에 있어서 가감산 연산을 적용하기 위하여 두 수를 $S_x X$와 $S_y Y$라고 할 때 S_x와 S_y은 각각 두 수의 부호로서 한 bit 이고, X와 Y는 각각 수의 크기이며 n bit 라고 하자. 두수의 가감산, 즉 $(S_x X)\pm(S_y Y)$에 대하여 수학적 결과, 실제로 행해야 할 연산, 최종 부호화-절대값 표현법, 그리고 기타 주의 사항들을 검토하기 위하여 표 1과 같이 진리표를 작성하였다. 여기서 Op는 $\pm$연산의 종류를 코드화한 것으로 0일 경우 +, 1일 경우 -를 뜻 한다.

표 1. $(S_x X)\pm(S_y Y)$ 연산
Table 1. $(S_x X)\pm(S_y Y)$ Arithmetic.

Op	S_x	S_y	수학적 결과	대소 비교	S-M 결과		비교
					S	mag.	
0	0	0	X+Y		0	X+Y	ovr
0	0	1	X-Y=-(Y-X)	Y<X	0	X-Y	
				X<Y	1	Y-X	
0	1	0	-X+Y=-(X-Y)	X>=Y	1	X-Y	
				X<Y	0	-X+Y	
0	1	1	-X-Y=-(X+Y)		1	X+Y	ovr
1	0	0	X-Y=-(Y-X)	X>=Y	0	X-Y	
				X<Y	1	Y-X	
1	0	1	X+Y		0	X+Y	ovr
1	1	0	-X-Y=-(X+Y)		1	X+Y	ovr
1	1	1	-X+Y=-(X-Y)	X>=Y	1	X-Y	
				X<Y	0	-X+Y	

이 표에서 몇 가지의 주제를 설명할 필요가 있다. 코드 $OpS_x S_y$가 000, 011, 101, 110에 대하여는 필요한 연산이 X+Y이며, 부호는 경우에 따라 + 혹은 -이다. 이 경우 주의할 점은 두 수의 합산 결과가 n bit의 표현범위를 벗어나는 오버플로우가 있으며 이는 carry 비트로 나타난다. 코드 $OpS_x S_y$가 001에 대하여 수학적 결과는 X-Y=-(Y-X)가 될 것이지만, 필요 연산과 최종 결과는 X와 Y의 대소 비교에 의하여 달라진다. 즉 $X\geq Y$인 경우의 연산은 X-Y을 수행해야 하고 최종 결과는 +(X-Y)가 된다. 반면에 $X<Y$인 경우는 연산은 Y-X을 수행해야 하고, 최종 결과는 -(Y-X)가 된다. 코드 $OpS_x S_y$가 010, 100, 111인 경우에도 동일한 해설이 가능하다.

개념적으로 위의 진리표에 의한 연산을 실현하기 위해서는 필요한 연산을 선택해야 하는데, 이때 X와 Y의 대소를 비교하는 비교기가 선행해야 한다. 그리고 선택해야 하는 연산의 종류는 X+Y, X-Y, Y-X와 같이 총 3종류이나, 마지막 두 개는 각각 X+(-Y)과 (-X)+Y로 변형할 수 있으므로 실제 필요한 연산은 합산만이며, 각각의 입력단자에 선택적인 2의 보수회로를 추가하여 X 혹은 -X, 그리고 Y 혹은 -Y를 선택하도록 해야 한다. 종합하면 대소 비교, 두 개의 선택적 2의 보수회로, 그리고 일반 가산기 회로가 필요할 것이다. 2의 보수는 각각의 비트의 보수를 취하여 얻는 1의 보수에 1을 더하면 되며, 이 경우는 언제나 둘 중 하나만 작용하므로, 그리고 뒤 따르는 가산기의 최 하단 carry 입력단자를 이용하여 1을 더하는 필요를 충족할 수 있으므로, 선택적 2의 보수회로 대신에 선택적 1의 보수회로로 간략히 할 수 있다.

이제까지의 개념적 설명에서 연산은 첫 번째 수에서 두 번째 수를 더하거나 빼는 일방적 가감 연산인 $(S_x X)\pm(S_y Y)$ 이었다. 그러나 부호화-절대값 표현법의 수 법칙을 이용하는 범용의 연산기를 위한 명령어 세트를 설계할 경우, 필요한 연산들 중에 두 수 모두에 가감산을 허용하는 연산, 즉 $\pm(S_x X)\pm(S_y Y)$이 더욱 포괄적일 수 있다. 통합적인 연산을 허용하더라도 두수의 절대치에 대한 필요한 기본 연산은 X+Y, X-Y, Y-X의 세 종류 가 될 것이므로, 본 논문에서는 제어부의 디코딩 로직이 조정되는 것 외에는 데이터 패스 부분의 하드웨어의 추가가 없이 부호화-절대값 표현법 수에 대한 통합적인 연산 $\pm(S_x X)\pm(S_y Y)$을 수행할 수 있는 부호화-절대값 가/감산기를 설계한다.

Ⅱ. 본 론

1. 두 수의 차이를 구하는 회로 설계

부호화-절대값 표현법의 수를 가감산하는 회로에 핵심적으로 필요한 두 수 X와 Y의 차이를 구하는 회로를 먼저 설계해 보자. 두 수의 차이는 $Y<X$인 경우는 $X-Y$이 될 것이고, $Y \geq X$인 경우는 $Y-X$가 된다. 명시적으로 X와 Y의 대소 비교를 먼저 수행한다면 이 작업은 개념적으로는 간단하지만, 이 작업은 첫 단에 두 수의 대소 비교 회로가 필요하고, 각각의 입력 단에 선택적 2의 보수회로가 추가되어야 하며, 그리고 가산기가 필요하므로 회로 구현상으로는 효과적이 아니다. 선행적인 대소 비교가 없이도 두 수의 차이를 구할 수 있도록 1차적으로 실행하는 연산은 X 입력단의 선택적 1의 보수를 행하고 carry 입력을 1로 하여 Y에 X의 2의 보수를 합산, 즉 $Y+\overline{X}+1$의 행한다. 이 합산의 결과로 나오는 carry인 C_{out}의 유무에 따라서 X와 Y의 대소를 판단할 수 있음이 다음과 같이 설명된다.

임의의 n bit 수 X에 X의 모든 비트를 각각 반전시킨 $\overline{X}$를 더하면, X에 관계없이, 다음의 항등식이 성립한다.

$$X+\overline{X}=2^n-1 \tag{1}$$

이 항등식을 재배열한 후, 양변에 Y를 합한 결과는 각각 다음과 같다.

$$\overline{X}+1=2^n-X \tag{2}$$

$$Y+\overline{X}+1=2^n+(Y-X) \tag{3}$$

위 식은 항등식이므로 좌변과 우변의 합산은 carry와 합에 있어서 각각 같은 결과를 주어야 한다. 위 식의 좌변을 n자리의 2진 연산으로 수행할 경우 n자리의 합은 2^n-1까지를 수용하며 발생하는 carry는 2^n의 값을 가진다. 좌변의 합산에서 carry가 나오면, 좌번과 같은 우변의 식은 $2^n+(Y-X) \geq 2^n$이 된다는 의미이고, 이를 정리하면 $Y \geq X$인 경우이다. 이 때 합산 $Y+\overline{X}+1$을 한 결과는 carry (2^n) 와 sum으로 두 수의 차이인 $Y-X$가 된다.

만약 좌변의 $Y+\overline{X}+1$ 합산에서 carry가 나오지 않으면 우변의 합산에서도 carry가 나오지 않아야 하는데, 이 경우는 $2^n+(Y-X)<2^n$ 즉 $Y<X$인 경우이다. 연산에서 carry가 발생하지 않은 경우에는 $Y<X$는 것이 확인되었으며, 따라서 구하는 두 수의 차이는 $X-Y$임을 알 수 있다. 이 때 합산 $Y+\overline{X}+1$을 한 결과는 carry는 없고 합은 $2^n+(Y-X)$이므로, 이 값으로부터 $X-Y$을 유도해야 한다. 위의 설명에서와 마찬가지로, 임의의 수 $2^n+(Y-X)$와 이 수의 1의 보수 $\overline{2^n+(Y-X)}$를 합산하면 항상 2^n-1이 되며, 이를 재배열 하면 원하는 결과가 얻어 진다. 즉 합산의 결과인 $2^n+(Y-X)$의 2의 보수를 구하면 원하는 두 수의 차이 $X-Y$가 된다:

$$\begin{aligned}\overline{2^n+(Y-X)}+\ (2^n+(Y-X)) &= 2^n-1\\ \overline{2^n+(Y-X)}+1 &= X-Y\end{aligned} \tag{4}$$

위의 두 경우를 종합하여 두수의 X와 Y의 차이를 구하기 위해서는 Y에 X의 2의 보수를 더하고, carry가 발생하면 두수의 대소는 $Y \geq X$이며 합산의 결과는 $Y-X$로서 두수의 차이가 된다. 만약 합산에서 carry가 발생하지 않으면 두수의 대소는 $Y<X$이며 합산의 결과는 $2^n+(Y-X)$이지만 원하는 두수의 차이인 $X-Y$는 $2^n+(Y-X)$의 2의 보수를 취함으로 얻을 수 있다.

2. 부호화-절대값 가/감산기

부호화-절대값 표현법의 통합 가감 연산 $\pm(S_xX)\pm(S_yY)$에서 S_x와 S_y 대신에 $\pm$를 치환하면 $\pm(\pm X)\pm(\pm Y)$가 되므로, 먼저 각수의 부호와 각수의 연산을 수학적으로 통합할 필요가 있다. 연산 기호를 O_x와 O_y로 표시하고, 이 기호가 해당 수의 가산과 감산을 의미하도록 각각 0과 1로 인코딩하자. 표 2는 각 수에 붙은 연산과 부호 변수 즉 O_xS_x와 O_yS_y을 수학적으로 각각 한 개의 변수 OS_x와 OS_y로 통합한 연산을 정의하는 진리표인데, 이 표가 의미하는 것은, 예를 들어 $+(-X)$과 $-(-Y)$이 각각 $-X$와 $+Y$로 수학적으로 통합된다는 것이다. 통합 연산을 식으로 간략히 하면 각각 $OS_x=O_x \oplus S_x$과 $OS_y=O_y \oplus S_y$으로 XOR 함수가 됨을 알 수 있다.

포괄적 연산 $\pm(\pm X)\pm(\pm Y)$에서 부호와 연산의 모든 가능성에 대하여 수학적 간략화를 하면 최종적인 부호

표 2. 부호와 연산의 통합

Table 2. Combination of Sign and Operation.

O_x (O_y)	S_x (S_y)	OS_x (OS_y)
0 +	0 +	0 +
0 +	1 -	1 -
1 -	0 +	1 -
1 -	1 -	0 +

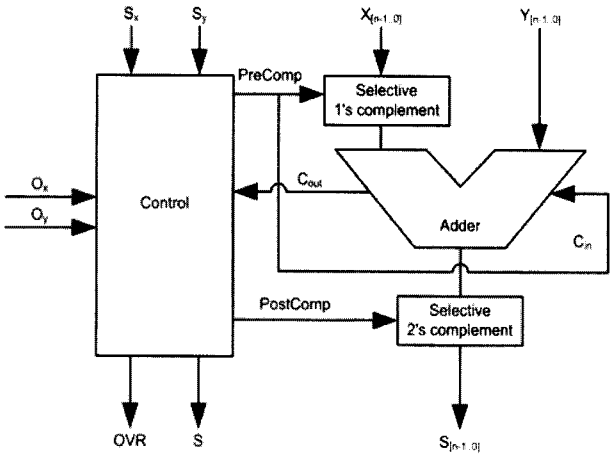

그림 1. 부호화-절대값 가/감산기 구조
Fig. 1. Signed-Magnitude Adder/Subtracter Structure.

화-절대값 표현의 결과 $\pm(X+Y)$와 $\pm|Y-X|$의 네가지 경우이다. 그러나 이 네 경우도 부호를 제외하고 결과 크기만을 고려하면 X와 Y의 합과 차이를 구하는 것으로 귀결된다. 2.1절에서 비교기 없이 두 수의 차이를 구하는 방법을 기술하였으므로, 이를 이용하여 본 논문에서 설계하는 비교기를 사용하지 않는 부호화-절대값 가/감산기의 구조는 그림 1과 같이 된다. 이 설계는 위에서 개념적으로 설명한 회로에서 대소 비교기를 제거하고, X 입력에만 선택적 2의 보수회로를 필요로 하며, 뒤따르는 가산기의 합산 결과로 두 수의 대소 비교를 추론하고, 그 추론 결과에 따라 필요하다면 합산 결과를 조정하기 위한 후단의 2의 보수 회로를 필요로 한다.

그림 1의 연산 구조에 의하여 그리고 통합 연산 변수를 사용하여 $\pm(S_x X)\pm(S_y Y)$의 연산을 위한 진리표는 표 3과 같으며, 각 열에 대한 상세한 해설과 이론적 분석이 따른다.

두 개의 통합 연산 변수에 따라 $\pm(S_x X)\pm(S_y Y)$의 연산은 4가지 경우로 나누어지며, 각각의 경우에 따라 수학적 결과는 쉽게 작성할 수 있다. 예를 들어 $O_x O_y = 01$의 경우는 수학적으로 X-Y를 구해야 하는데 이는 수학적으로 -(Y-X)와도 같음을 표시하였다. 두 통합 연산 변수가 서로 같은 경우, $O_x O_y = 00, 11$, 연산의 수학적 결과는 두 절대치의 합을 구해야 함으로, 1의 보수 기능은 선택되지 않으며 (PreComp=0), 가산기의 입력 carry는 0이 된다 ($C_{in} = 0$). 절대치의 합산의 결과에 따르는 최종 부호는 통합 연산의 종류에 따라 달라진다. 통합 연산이 모두 합산이면 부호는 +이고, 반대로 통합 연산이 모두 감산이면 부호는 -가 된다 (S-M 결과 부호). 주의할 점은 두 수의 합산 과정에 carry ($C_{out} = 1$)가 나오면 오버플로우가 생긴 경우이며 (ovr), carry를 제외한 연산 결과는 $X+Y-2^n$이 된다.

두 통합 연산 변수가 서로 다른 경우, $O_x O_y = 01, 10$, 연산의 수학적 결과는 X-Y 혹은 Y-X이지만, 어느 경우나 1차적으로 두 수의 차이를 구해야 한다. 2.1절에서 설명한 것과 같이 두 수 X와 Y의 차이를 구하기 위해서는 Y에 X의 2의 보수를 합산 즉 즉 $Y+\overline{X}+1$의 연산을 행해야 한다. 이를 위하여 X의 2의 보수를 구해야 함으로 PreComp=1이 된다. 합산 과정에 carry가 발생하면 (C_{out}=1), 두 수의 대소는 $Y \geq X$이며 합산의 결과는 Y-X로서 두 수의 차이가 된다. 만약 합산에서 carry가 발생하지 않으면 (C_{out}=0), 두 수의 대소는 $Y < X$이며 합산의 결과는 $2^n+(Y-X)$이지만 원하는 두 수의 차이인 X-Y는 $2^n+(Y-X)$의 2의 보수를 취함으로 얻을 수 있다 (PostComp=1). 이것이 표 3에서 두 수의 차이를 구하는 경우($O_x O_y = 01, 10$)에, 합산에서 carry가 나오지 않는 경우 PostComp=1이 되게 한 이유이다.

표 3에서 S-M(부호화-절대값 : singed- magnitude) 결과의 부호와 절대치 열은 각각의 통합연산 변수에 따

표 3. $\pm(S_x X)\pm(S_y Y)$ 연산
Table 3. $\pm(S_x X)\pm(S_y Y)$ Arithmetic.

OS_x	OS_y	수학적 결과	수행연산	PreComp (C_{in})	C_{out}	대소비교 (ovr)	연산결과	S-M 결과		PostComp
								부호	절대치	
0	0	X+Y	Y+X	0	0		Y+X	0	Y+X	0
					1	ovr	$Y+X-2^n$	0	$Y+X-2^n$	0
0	1	X-Y=-(Y-X)	$Y+\overline{X}+1$	1	0	Y<X	$Y-X+2^n$	0	X-Y	1
					1	Y≥X	Y-X	1	Y-X	0
1	0	Y-X=-(X-Y)	$Y+\overline{X}+1$	1	0	Y<X	$Y-X+2^n$	1	X-Y	1
					1	Y≥X	Y-X	0	Y-X	0
1	1	-(X+Y)	Y+X	0	0		Y+X	1	Y+X	0
					1	ovr	$Y+X-2^n$	1	$Y+X-2^n$	0

른 수학적 결과, 대소비교, 그리고 연산결과에 따라 정할 수 있다. 예를 들어 두 통합 연산이 $O_x O_y = 10$ 인 경우, 수학적 결과는 $Y-X=-(X-Y)$이 되어야 한다. 이를 위하여 $Y+\overline{X}+1$의 연산을 한 후에 carry가 발생하면, 대소 비교는 $Y \geq X$이 되고, 연산결과는 $Y-X$가 된다. 수학적 결과와 대소 비교를 검토하여 최종 S-M 결과는 $+(Y-X)$가 된다. 따라서 부호는 0 즉 양수가 된다. 반면에 carry가 발생하지 않으면, 대소비교가 $Y<X$이 되고, 연산결과는 $2^n+(Y-X)$가 된다. 수학적 결과와 대소 비교를 검토하여 최종 S-M 결과는 $-(X-Y)$ 된다. 따라서 부호는 1 즉 음수가 된다. 통합 연산 변수의 다른 경우에 대해서도 동일한 해석을 할 수 있으므로 부호와 절대치 column을 표 2에 주어진 것 같이 규정할 수 있다.

표 3의 진리표에 의하여 제어부에 필요한 제어신호와 부호, 오버플로우 등의 상태 신호는 다음과 같이 디코딩 할 수 있다:

$$\text{PreComp} = C_{in} = OS_x \oplus OS_y = O_x \oplus S_x \oplus O_y \oplus S_y$$

$$\text{PostComp} = \text{PreComp} \cdot \overline{C_{out}} = (O_x \oplus S_x \oplus O_y \oplus S_y) \cdot \overline{C_{out}}$$

$$\text{OVR} = \overline{\text{PreComp}} \cdot C_{out} = \overline{(O_x \oplus S_x \oplus O_y \oplus S_y)} \cdot C_{out}$$

$$\begin{aligned}\text{Sign} &= OS_x \cdot \overline{OS_y} + \overline{OS_x} \cdot OS_y \cdot C_{out} + OS_x \cdot \overline{OS_y} \cdot \overline{C_{out}} \\ &= OS_X \cdot \overline{C_{out}} + OS_y \cdot C_{out}\end{aligned}$$

Ⅲ. 구현 및 검증

표 3의 진리표에 의한 두개의 통합 연산 변수에 따른 제어부는 그림 2로 구현된다.

본 논문에서 설계한 부호화-절대값 가/감산기는 Altera사의 MAX+plus Ⅱ를 이용하여 FPGA(FLEX 10K EPF10K20RC240-3)로 구현하여 동작을 검증하였다. 가/감산기의 기능을 검증하기 위하여 임의의 두 값을 더하여 올바른 결과가 나오는 지를 확인하였고, 가/감산기의 효용성(회로의 면적, 지연시간)은 Report파일과 Timing Analysis를 통해 사용된 LC의 개수와 최고 지연시간을 기준으로 비교 분석하였다.

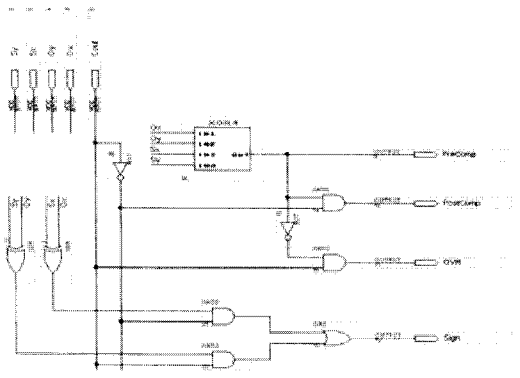

그림 2. 제어부
Fig. 2. Control Block.

표 4. 하드웨어와 지연시간 비교
Table 4. Comparison of Hardware & Time Delay

	하드웨어(LCs)	최고 지연시간 (ns)
a. 비교기를 사용한 방식	174	126.9
b.본 논문에서 제안한 방식	144	129.6

a. 비교기를 사용한 방식

```
** DEVICE SUMMARY **

Chip/                  Input Output Bidir  Memory  Memory                    LCs
POF      Device        Pins  Pins   Pins   Bits %  Utilized  LCs  % Utilized

32addsub_comp2
      EPF10K20RC240-3  65    34     0      0         0 %     174     15 %

User Pins:             65    34     0
```

b. 본 논문에서 제안한 방식

```
** DEVICE SUMMARY **

Chip/                  Input Output Bidir  Memory  Memory                    LCs
POF      Device        Pins  Pins   Pins   Bits %  Utilized  LCs  % Utilized

32addsub_top
      EPF10K20RC240-3  66    36     0      0         0 %     144     12 %

User Pins:             66    36     0
```

제어부의 신호들에 의하여 의도된 부호화-절대값 연산이 수행되는 것을 부호와 크기를 포함하여 5 비트인 10101 (-5)와 01101 (+13)인 두 수의 통합 연산 $-10101 - 01101$에 대하여 시험하여 보자. 주어진 데이터에서 $O_x = 1$, $S_x = 1$, $X = 0101(3)$, $O_y = 1$, $S_y = 0$, $Y = 1101(13)$이 분리 된다. Pre-complement 신호 $\text{PreComp} = C_{in} = 1 \oplus 1 \oplus 1 \oplus 0 = 1$이 되므로 수행하는 연산은 $Y+\overline{X}+1$이며, 각각의 값을 대입하면 $1101 + 1010 + 1 = \{C_{out} = 1 \ \& \ \text{sum } 1000\}$이 된다. 연산에서 carry가 나오므로 $Y \geq X$로 판별할 수 있으며, 연산 결과는 10진수로 8인데, 이는 모두 주어진 데이터로서 확인할 수 있다. 또한 $\text{PostComp} = \text{PreComp} \cdot \overline{C_{out}} = 1 \cdot \overline{1} = 0$이 되어 가산의 결과의 2의 보수를 취할 필요가 없이 바로 최종값이 되며, 이는 주어진 데이터에 일치한다. 최종 부호는 $\text{Sign} = OS_X \cdot \overline{C_{out}} + OS_y \cdot C_{out} = (1 \oplus 1) \cdot \overline{1} + (1 \oplus 0) \cdot 1 = 1$이 되어 음수가 되는데, 마찬가지로 주어진 데이터의 통합 연산과 일치한다.

그림 3은 설계한 회로의 검증 결과를 보여주고 있다.

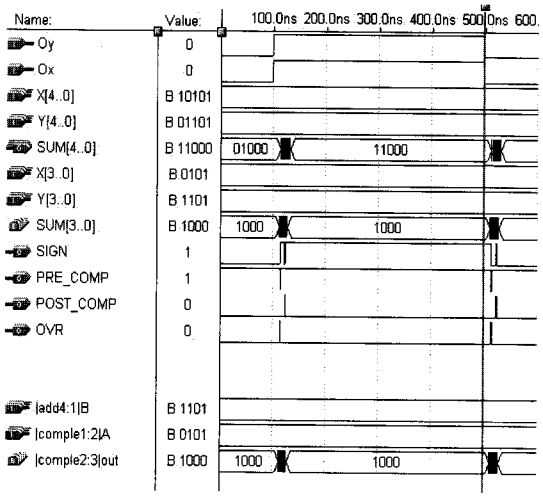

그림 3. 검증결과 : -(5) - (+13) = -8
Fig. 3. Simulation Result : -(5) - (+13) = -8.

Ⅳ. 결 론

부호화-절대값으로 표현되는 두 수 (±A)와 (±B)의 ±(±A)±(±B) 형태의 통합 연산을 처리함에 있어서 두 수의 대소비교가 선행적으로 이루어져야 하는데, 본 논문에서는 부호화-절대값 표현 수에 대해서 명시적인 비교기 회로를 사용하지 않고 두 수의 차이를 구하는 회로를 설계하였으며, 이 회로를 이용하여 일반화된 연산 즉 ±(±X)±(±Y)을 수행하는 가/감산기를 설계하였다. 본 논문에서 제안한 가/감산기의 제어부는 처리하는 두 수의 워드 크기(비트수)에 영향을 받지 않고 워드 크기가 증가하더라도 동일하다. 처리하는 데이터 크기가 커질 경우에 데이터 패스의 각 부분들만 독립적으로 연결하면 된다. 따라서 처리하는 수의 워드 크기가 커질수록 하드웨어와 시간지연이 커지는 기존의 비교기를 사용한 방식에 비해서 효율적인 방식이다.

본 논문에서는 기존의 방식에서 비교기 대신에 두 수의 사인 비트와 수행하는 연산작업에 의해서 두 수의 크기비교를 추론하는 간단한 구조의 제어기를 사용함으로서 비슷한 지연시간을 가지면서 기존의 비교기를 사용하는 방식에 비해서 17%의 하드웨어를 절약할 수 있었다.

참 고 문 헌

[1] Parhami, Behrooz "Computer Arithmetic: Algorithms and Hardware Designs", Oxford Univ. Press, Inc. Oxford New York, 2000. pp 19-28
[2] Koren, Israel, "Computer Arithmetic Algorithms", A.K. Peters, Ltd., Natick, MA, 2002. pp 6-14
[3] Kantabutra, V., "A Recursive Carry-Lookahead /Carry-Select Hybrid Adder," IEEE Trans. Computers, Vol. 42, No. 12, pp. 1495-9, 1993
[4] Kai Hwang, "Computer Arithmetic : Principle, Architecture, and Design", John Wiley & Sons Ltd. 1979. pp 69-71, 107-112
[5] Ivan Flores, "The Logic of COMPUTER ARITHMETIC", Prentice-Hall, Inc. 1963. pp 20-43
[6] John B. Gosling, "Design of Arithmetic Units for Digital Computers", Springer-Verlag New York Inc. 1980. pp 39-54

저 자 소 개

정 태 상(정회원)
1978년 서울대학교 전기공학과 (학사)
1982년 미국 Ohio 주립대 (석사)
1985년 미국 Ohio 주립대 (박사)
1986년~1992년 미국 Kentucky대 조교수
1992년~현재 중앙대학교 전자전기공학부 교수
<주관심분야 : 컴퓨터, 반도체, SoC 설계>

권 금 철(학생회원)
2001년 중앙대학교 전자전기 공학부 학사 졸업.
2003년 중앙대학교 전자전기 공학부 석사 졸업.
2007년~현재 중앙대학교 전자 전기공학부 박사 과정
<주관심분야 : 컴퓨터, 반도체, SoC 설계>